

1           **1. (original)**   A finite state model-based testing system comprising:  
2           a model generation engine to generate a model of a software application to  
3           be tested; and  
4           a graphical user interface to enable user entry of parameters for defining the  
5           model.

6  
7           **2. (original)**   A finite state model-based testing system as recited in  
8           claim 1, wherein the user interface enables a user to enter state information and  
9           transition information about the software application, the transition information  
10          describing a next state of the software application after an input has been applied  
11          to a current state of the software application.

12  
13          **3. (original)**   A finite state model-based testing system as recited in  
14          claim 1, wherein the user interface enables a user to enter state information about  
15          the software application, the state information comprising:

16          an operational mode of the software application, wherein the operational  
17          mode is an attribute of a particular state of the software application;

18          at least one modal value associated with the operational mode, wherein the  
19          modal value describes a behavior of the operational mode; and

20          an input of the software application.  
21  
22  
23  
24  
25

1           **4. (original)**   A finite state model-based testing system as recited in  
2 claim 1, wherein the user interface enables a user to enter transition information  
3 about the software application, the transition information comprising:

4               a current state of the software application, the current state being associated  
5 with an input of the software application; and

6               a next state of the software application, the next state indicating the state of  
7 the software application after the input has been applied to the current state of the  
8 software application.

9  
10           **5. (original)**   A finite state model-based testing system as recited in  
11 claim 1, wherein the user interface comprises a model editor to enable user entry  
12 of an operational mode, a modal value associated with the operational mode, and  
13 an input of the software application for defining the model.

14  
15           **6. (original)**   A finite state model-based testing system as recited in  
16 claim 1, wherein the user interface comprises a rules editor to enable user entry of  
17 an input of the software application, a current state of the software application, and  
18 a next state of the software application indicating the state of the software  
19 application after the input has been applied to the current state of the software  
20 application for defining the model.

1           7.    **(original)**   A finite state model-based testing system as recited in  
2 claim 1, wherein the model of the software application is a state table, the state  
3 table having at least one state table entry, and wherein:

4           a state table entry comprises:

5               (1) a current state of the software application;

6               (2) an input of the software application;

7               (3) a next state of the software application, the next state indicating  
8 the state of the software application after the input has been applied to the  
9 current state of the software application;

10          the model generation engine evaluates the current state of the software  
11 application to determine if an input of the software application can be applied to  
12 the current state and in the event that the input can be applied to the current state,  
13 writes a state table entry out to the state table.  
14

15          8.    **(original)**   A finite state model-based testing system as recited in  
16 claim 1, wherein the user interface comprises a graph traversal menu to enable a  
17 user to select a graph traversal program and generate a test sequence of inputs for  
18 the software application.  
19

20          9.    **(original)**   A finite state model-based testing system as recited in  
21 claim 1, further comprising a graph traversal program to generate a test sequence  
22 of inputs for the software application, the test sequence of inputs generated from  
23 the model of the software application with the graph traversal program.  
24  
25

1           **10. (original)**   A finite state model-based testing system as recited in  
2 claim 1, wherein the user interface comprises a test execution menu to enable a  
3 user to select a test driver program and initiate a test of the software application.

4  
5           **11. (original)**   A finite state model-based testing system as recited in  
6 claim 1, further comprising a test driver program to execute a test sequence of  
7 application inputs on the software application.

8  
9           **12. (original)**   A user interface for testing a software application, the  
10 user interface comprising:

11           a model editor to enable user entry of state information to define a model of  
12 a software application to be tested; and

13           a rules editor to enable user entry of transition information to further define  
14 the model of the software application to be tested, the transition information  
15 describing a next state of the software application after an input has been applied  
16 to a current state of the software application.

17  
18           **13. (original)**   A user interface as recited in claim 12, wherein the  
19 user interface is a graphical user interface.

1       **14. (original)**   A user interface as recited in claim 12, wherein the  
2 state information comprises:

3       an operational mode of the software application, wherein the operational  
4 mode is an attribute of a particular state of the software application;

5       at least one modal value associated with the operational mode, wherein the  
6 modal value describes a behavior of the operational mode; and

7       an input of the software application.

8  
9       **15. (original)**   A user interface as recited in claim 12, wherein the  
10 transition information comprises:

11       a current state of the software application, the current state being associated  
12 with an input of the software application; and

13       a next state of the software application, the next state indicating the state of  
14 the software application after the input has been applied to the current state of the  
15 software application.

16  
17       **16. (original)**   A user interface as recited in claim 12, wherein the  
18 model editor comprises:

19       an operational modes entry field to enable user entry of an operational  
20 mode of the software application; and

21       an operational modes list field to display the operational mode.

1       **17. (original)**   A user interface as recited in claim 12, wherein the  
2 model editor comprises:

3           a modal values entry field to enable user entry of a modal value associated  
4 with an operational mode of the software application; and

5           a modal values list field to display the modal value.  
6

7       **18. (original)**   A user interface as recited in claim 12, wherein the  
8 model editor comprises:

9           an inputs entry field to enable user entry of an input of the software  
10 application; and

11           an inputs list field to display the input of the software application.  
12

13       **19. (original)**   A user interface as recited in claim 12, wherein the  
14 rules editor comprises fields to display the state information that can be entered  
15 using the model editor, the fields comprising:

16           inputs fields to display inputs of the software application, wherein the  
17 inputs fields also enable user selection of an input of the software application;

18           operational modes fields to display operational modes of the software  
19 application, wherein the operational modes fields also enable user selection of an  
20 operational mode; and

21           modal values fields to display modal values associated with an operational  
22 mode, wherein the modal values fields also enable user selection of a modal value.  
23  
24  
25

1       **20. (original)**   A user interface as recited in claim 19, wherein the  
2 rules editor enables user entry of the transition information, and wherein:

3       the transition information comprises:

4               a current state of the software application, the current state being  
5 associated with the input of the software application;

6               a next state of the software application, the next state indicating the  
7 state of the software application after the input has been applied to the  
8 current state of the software application;

9       the rules editor comprises:

10              an inputs field to enable user entry of the input;

11              a current state operational mode field to enable user entry of the  
12 operational mode as a current state operational mode;

13              a current state modal value field to enable user entry of the modal  
14 value associated with the current state operational mode;

15              a next state operational mode field to enable user entry of the  
16 operational mode as a next state operational mode; and

17              a next state modal value field to enable user entry of the modal value  
18 associated with the next state operational mode.

1           **21. (original)**   A user interface as recited in claim 12, wherein the  
2 model is a state table having at least one state table entry, the state table entry  
3 having a current state of the software application, an input of the software  
4 application, and a next state of the software application, the next state indicating  
5 the state of the software application after the input has been applied to the current  
6 state of the software application; and

7           the model editor enables user initiation of a model generation engine to  
8 generate the model of the software application, the model generation engine being  
9 configured to evaluate a current state of the software application to determine if an  
10 input of the software application can be applied to the current state and in the  
11 event that the input can be applied to the current state, writes a state table entry out  
12 to the state table.

13  
14           **22. (original)**   A user interface as recited in claim 12, wherein the  
15 model editor enables user initiation of a graph traversal program to generate a test  
16 sequence of inputs for the software application.

17  
18           **23. (original)**   A user interface as recited in claim 12, wherein the  
19 user interface further comprises a graph traversal menu to enable user selection of  
20 a graph traversal program to generate a test sequence of inputs for the software  
21 application.



1           **24. (original)**   A user interface as recited in claim 23, wherein the  
2 graph traversal menu comprises:

3           a graph traversal program field to enable user selection of the graph  
4 traversal program;

5           a model file field to enable user selection of the model of the software  
6 application to be tested; and

7           a test sequence file field to enable user entry of a memory storage location  
8 for a test sequence file, the test sequence file containing the test sequence of inputs  
9 for the software application.

10  
11           **25. (original)**   A user interface as recited in claim 12, wherein the  
12 model editor enables user initiation of a test driver program to read a test sequence  
13 of inputs for the software application and apply the test sequence to the software  
14 application.

15  
16           **26. (original)**   A user interface as recited in claim 12, wherein the  
17 user interface further comprises a test execution menu to enable user selection of a  
18 test driver program to read a test sequence of inputs for the software application  
19 and apply the test sequence to the software application.

1           **27. (original)**   A user interface as recited in claim 26, wherein the test  
2 execution menu comprises:

3           a test driver program field to enable user selection of the test driver  
4 program;

5           a test sequence file field to enable user selection of a test sequence file  
6 containing the test sequence of inputs for the software application to be tested; and

7           a test monitoring interval field to enable user entry of a timing interval to  
8 define how often the test driver program can be monitored to detect a failure of the  
9 test driver program.

10  
11           **28. (original)**   A user interface to enable user entry of parameters to  
12 define a model of a software application to be tested, the user interface  
13 comprising:

14           an operational modes field to enable user entry of an operational mode of  
15 the software application, the operational mode being an attribute of a particular  
16 state of the software application; and

17           a modal values field to enable user entry of at least one modal value  
18 associated with the operational mode, the modal value describing a behavior of the  
19 operational mode.

20  
21           **29. (original)**   A user interface as recited in claim 28, further  
22 comprising an input field to enable user entry of an input of the software  
23 application.

1       **30. (original)** A user interface as recited in claim 29, further  
2 comprising:

3       an operational modes list field to display the operational mode;

4       a modal values list field to display the modal value; and

5       an inputs list field to display the input of the software application.

6  
7       **31. (original)** A user interface as recited in claim 28, wherein the  
8 user interface enables user initiation of a graph traversal program to generate a test  
9 sequence of inputs for the software application.

10  
11       **32. (original)** A user interface as recited in claim 28, wherein the  
12 user interface enables user initiation of a test driver program to read a test  
13 sequence of inputs for the software application and apply the test sequence to the  
14 software application.

15  
16       **33. (original)** A user interface to enable user entry of parameters to  
17 define a model of a software application to be tested, the user interface  
18 comprising:

19       an inputs field to enable user entry of an input of the software application;

20       current state fields to enable user entry of a current state of the software  
21 application, the current state being associated with the input; and

22       next state fields to enable user entry of a next state of the software  
23 application, the next state indicating the state of the software application after the  
24 input has been applied to the current state of the software application.

1       **34. (original)**   A user interface as recited in claim 33, further  
2 comprising a rules field to enable user entry of a rule to describe a transition of the  
3 software application from a current state to a next state.

4  
5       **35. (original)**   A user interface as recited in claim 34, further  
6 comprising a control to enable a user to disable a rule such that the model of the  
7 software application will be defined without the rule.

8  
9       **36. (original)**   A user interface as recited in claim 33, wherein:

10 the current state fields include:

11       a current state operational mode field to enable user entry of a  
12 current state operational mode of the software application;

13       a current state modal value field to enable user entry of at least one  
14 current state modal value associated with the current state operational  
15 mode;

16 the next state fields include:

17       a next state operational mode field to enable user entry of a next  
18 state operational mode of the software application; and

19       a next state modal value field to enable user entry of at least one next  
20 state modal value associated with the next state operational mode.  
21  
22  
23  
24  
25

1       **37. (original)**     A user interface as recited in claim 36, wherein:

2       the current state fields include:

3               a current state relational operator field to indicate the current state  
4       modal value relation to the current state operational mode;

5               a current state concatenation operator field to indicate the relation  
6       between a first current state rule criteria and a second current state rule  
7       criteria.

8       the next state fields include:

9               a next state relational operator field to indicate the next state modal  
10      value relation to the next state operational mode; and

11              a next state concatenation operator field to indicate the relation  
12      between a first next state rule criteria and a second next state rule criteria.

13  
14       **38. (original)**     A data structure stored on a computer readable  
15      medium comprising:

16              at least one mode data structure to hold an operational mode of a software  
17      application to be tested and at least one modal value associated with the  
18      operational mode; and

19              at least one rule data structure to hold an input of the software application  
20      to be tested, a current state of a software application, and a next state of the  
21      software application.

22  
23       **39. (original)**     A data structure as recited in claim 38, wherein the  
24      mode data structure is a linked list of one or more mode data structures.

1           **40. (original)**   A data structure as recited in claim 38, wherein the  
2 rule data structure is a linked list of one or more rule data structures.

3  
4           **41. (original)**   A data structure as recited in claim 38, wherein:  
5           the current state and the next state are defined by rule criteria data  
6 structures; and  
7           the rule criteria data structures are stored in a linked list.

8  
9           **42. (original)**   A finite state model-based testing system comprising:  
10          a model editor to enable user entry of state information to define a model of  
11 a software application to be tested;  
12          a rules editor to enable user entry of transition information to further define  
13 the model of the software application, the transition information describing a next  
14 state of the software application after an input has been applied to a current state  
15 of the software application;  
16          a model generation engine to generate the model of the software  
17 application;  
18          a graph traversal menu to enable user selection of a graph traversal program  
19 to generate a test sequence of inputs for the software application; and  
20          a test execution menu to enable user selection of a test driver program to  
21 read the test sequence of inputs for the software application and apply the test  
22 sequence to the software application.

1           **43. (original)**   A finite state model-based testing system as recited in  
2 claim 42, wherein the model editor comprises:

3           operational modes fields to enable user entry of an operational mode of the  
4 software application;

5           modal values fields to enable user entry of a modal value associated with  
6 the operational mode; and

7           inputs fields to enable user entry of an input of the software application.  
8

9           **44. (original)**   A finite state model-based testing system as recited in  
10 claim 42, wherein the rules editor comprises fields to display the state information  
11 that can be entered using the model editor, the fields comprising:

12           inputs fields to display inputs of the software application, wherein the  
13 inputs fields also enable user selection of an input of the software application;

14           operational modes fields to display operational modes of the software  
15 application, wherein the operational modes fields also enable user selection of an  
16 operational mode; and

17           modal values fields to display modal values associated with an operational  
18 mode, wherein the modal values fields also enable user selection of a modal value.  
19  
20  
21  
22  
23  
24  
25

1           **45. (original)**   A finite state model-based testing system as recited in  
2 claim 42, wherein the model is a state table having at least one state table entry,  
3 the state table entry having a current state of the software application, an input of  
4 the software application, and a next state of the software application; and

5           the model editor enables user initiation of the model generation engine  
6 which is configured to evaluate a current state of the software application to  
7 determine if an input of the software application can be applied to the current state  
8 and in the event that the input can be applied to the current state, writes a state  
9 table entry out to the state table.

10  
11           **46. (original)**   A finite state model-based testing system as recited in  
12 claim 42, wherein the model editor facilitates user initiation of the rules editor.

13  
14           **47. (original)**   A finite state model-based testing system as recited in  
15 claim 42, wherein the model editor facilitates user initiation of the graph traversal  
16 menu.

17  
18           **48. (original)**   A finite state model-based testing system as recited in  
19 claim 42, wherein the model editor facilitates user initiation of the test execution  
20 menu.



1       **49. (original)**   A computer system comprising:

2       a processor;

3       a memory;

4       a user interface application stored in the memory and executable on the  
5 processor to facilitate user definition of a finite-state model to test a software  
6 application;

7       the user interface application having computer readable instructions to  
8 display a graphical user interface; and

9       a model generation engine stored in memory and executable on the  
10 processor to generate the model of the software application.

11  
12       **50. (original)**   A computer system as recited in claim 49, wherein the  
13 user interface enables a user to enter state information and transition information  
14 about the software application, the transition information describing a next state of  
15 the software application after an input has been applied to a current state of the  
16 software application.

17  
18       **51. (original)**   A computer system as recited in claim 49, wherein the  
19 user interface comprises a model editor to enable user entry of operational modes,  
20 modal values, and inputs of the software application to define the model.

1           **52. (original)**   A computer system as recited in claim 49, wherein the  
2 user interface comprises a rules editor to enable user entry of an input of the  
3 software application, a current state of the software application, and a next state of  
4 the software application to define the model.

5  
6           **53. (original)**   A computer system as recited in claim 49, further  
7 comprising at least one graph traversal program stored in the memory and  
8 executable on the processor to generate a test sequence of inputs for the software  
9 application, the user interface presenting a graph traversal menu to enable a user to  
10 select the graph traversal program.

11  
12           **54. (original)**   A computer system as recited in claim 49, further  
13 comprising at least one test driver program stored in the memory and executable  
14 on the processor to execute a test sequence of application inputs on the software  
15 application, the user interface presenting a test execution menu to enable a user to  
16 select the test driver program.

17  
18           **55. (original)**   A computer system as recited in claim 49, further  
19 comprising a data structure stored in the memory, the data structure comprising:

20           at least one mode data structure to hold an operational mode of a software  
21 application and at least one modal value associated with the operational mode; and

22           at least one rule data structure to hold an input of the software application, a  
23 current state of a software application, and a next state of the software application.  
24  
25

1           **56. (original)**   A method comprising:

2           presenting a graphical user interface that facilitates user entry of state  
3           information and transition information about a software application to be tested;  
4           and

5           generating a model of the software application using the state information  
6           and the transition information.

7  
8           **57. (original)**   A method as recited in claim 56, further comprising:

9           presenting a graphical user interface that facilitates user selection of a graph  
10          traversal program; and

11          generating a test sequence of inputs for the software application.

12  
13          **58. (original)**   A method as recited in claim 56, further comprising:

14          presenting a graphical user interface that facilitates user selection of a test  
15          driver program; and

16          executing a test sequence of application inputs on the software application.

17  
18          **59. (original)**   A method as recited in claim 56, further comprising

19          enabling a user to define a transition rule of the software application, wherein the  
20          enabling comprises presenting a graphical user interface to facilitate user entry of  
21          an input of the software application, a current state of the software application  
22          associated with the input, and a next state of the software application.

1           **60. (original)**   A method as recited in claim 56, further comprising:  
2           presenting a graphical user interface that facilitates a user defining a  
3           transition rule of the software application and disabling the transition rule; and  
4           generating the model of the software application without incorporating the  
5           disabled transition rule.

6  
7           **61. (original)**   A method as recited in claim 56, wherein generating a  
8           model of the software application comprises:  
9           evaluating a current state of the software application to determine if an  
10          input of the software application can be applied to the current state; and  
11          in the event that the input can be applied to the current state, writing a state  
12          table entry out to a state table.

13  
14          **62. (original)**   A method as recited in claim 56, further comprising  
15          enabling a user to define the state information, wherein the enabling comprises  
16          presenting a graphical user interface to facilitate user entry of an operational mode  
17          of the software application, at least one modal value associated with the  
18          operational mode, and an input of the software application.

1           **63. (original)**   A method as recited in claim 56, further comprising  
2 enabling a user to define the transition information, wherein the enabling  
3 comprises presenting a graphical user interface to facilitate user entry of a current  
4 state of the software application, the current state being associated with an input of  
5 the software application, and a next state of the software application, the next state  
6 indicating the state of the software application after the input has been applied to  
7 the current state of the software application.

8  
9           **64. (original)**   A computer-readable medium comprising computer  
10 executable instructions that, when executed, direct a computing system to perform  
11 the method of claim 56.

12  
13           **65. (original)**   A method comprising:  
14           presenting a user interface that facilitates user entry of state information and  
15 transition information about a software application to be tested;  
16           initiating, via the user interface, generation of a model of the software  
17 application;  
18           selecting, via the user interface, a graph traversal program that generates a  
19 test sequence of inputs for the software application; and  
20           selecting, via the user interface, a test driver program that executes a test  
21 sequence of application inputs on the software application.

1       **66. (original)**   A computer-readable medium comprising computer  
2       executable instructions that, when executed, direct a computing system to perform  
3       the method of claim 65.

4  
5       **67. (original)**   A method comprising:  
6       receiving state information about a software application to be tested;  
7       receiving transition information about the software application;  
8       generating a model of the software application;  
9       from the model, generating a test sequence of inputs for the software  
10      application with a graph traversal program; and  
11      executing a test sequence of application inputs on the software application.

12  
13      **68. (original)**   A method as recited in claim 67, wherein generating a  
14      model of the software application comprises:  
15      evaluating a current state of the software application to determine if an  
16      input of the software application can be applied to the current state; and  
17      in the event that the input can be applied to the current state, writing a state  
18      table entry out to a state table.

19  
20      **69. (original)**   A computer-readable medium comprising computer  
21      executable instructions that, when executed, direct a computing system to perform  
22      the method of claim 67.

1        **70. (original)**    A computer-readable medium comprising computer  
2 executable instructions that, when executed, direct a computing system to:  
3        receive state information about a software application to be tested;  
4        receive transition information about the software application;  
5        generate a model of the software application;  
6        from the model, generate a test sequence of inputs for the software  
7 application with a graph traversal program; and  
8        execute a test sequence of application inputs on the software application.